NPS-53-85-0007

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A VARIANCE DETECTOR FOR SIGNAL-GAP DISCRIMINATION IN NOISY SPEECH CHANNELS

by

James L. Wayman

May 1985

Technical Report For Period

August 1984 - September 1984

Approved for public release; distribution unlimited

Prepared for: The Naval Postgraduate School
Monterey, California 93943

NAVAL POSTGRADUATE SCHOOL
MONTEREY CALIFORNIA 93943

R. H. SHUMAKER
Rear Admiral, U. S. Navy
Superintendent

D. A. SCHRADY
Provost

This report was prepared by:

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NPS-53-85-0007	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): A Variance Detector for Signal-Gap Discrimination in Noisy Speech Channels	5. TYPE OF REPORT & PERIOD COVERED: Technical Report Interim, 8/84-9/84
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): James L. Wayman	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943	12. REPORT DATE: May 1985
	13. NUMBER OF PAGES: 20
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): N/A	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Signal/Gap detection

Digitial signal processing of speech

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report documents a program written for post- (FFT) processor activity/gap detection in noisy speech communication channels. The algorithm used is based on the assumption that the variance of the logs of the frequency-domain magnitudes of a signal is greater than the corresponding variance of the channel noise. Trade-offs between Type I and Type II errors can be controlled with an interactively defined program parameters. Parameters were found to yield a 16% missed detection rate with a 5% false alarm rate, performance entirely acceptable in speech compression applications. Some problems were encountered with

BLOCK 20 - ABSTRACT (CONTINUATION)

sibilate detection, due to the close relationship between these sounds and white noise.

VARIANCE DETECTOR REPORT

INTRODUCTION

This report documents the program "vardet.c" written in July, 1984 for post- (FFT) processor activity/gap detection. The algorithm used is based on the assumption that the variance of the logs of the frequency- domain magnitudes of a signal is greater than the corresponding variance of the channel noise.

The noise is assumed to be quasi-ergodic, that is ensemble and single FFT averages and variances are assumed to be the same, except that these measures are allowed to vary slowly in time using a weighted, recursive technique.

Trade-offs between Type I (false detection) and Type II (missed signal) errors can be controlled with an interactively defined program parameter.

The program is designed to warm-up gracefully, if an a priori estimate of an upper bound on the variance of the logs of the noise magnitudes is available. In the absence of such an estimate, the warm-up period can be extended interactively to allow the program to learn the system noise levels.

Additionally, this program allows the addition of noise to an input signal so that program parameters may be tested under various signal-to-noise ratios.

The output of the program is in two forms. Indication of activity or gap is printed to a file called "GRAF" for graphing on the line printer. For periods in which activity is detected, the raw, pre-FFT data is printed to a file called "ACTIVE" for D/A conversion to recover the gap-surpressed signal. An estimation of the signal-to-noise ratio is also available for printing.

TASK DEFINITION

The problem is to design an activity/gap detector for digitized signal data of the following three types: speech, fsk, and psk. Running time is limited by the specifications of the mode analysis unit into which the detector is to be incorporated, and number of operations is limited by hardware considerations. The measure of effectiveness can be given by the complement of the following function:

1) $\emptyset = C * \int \text{MISSED DETECTIONS} \; d(snr) + \int \text{FALSE ALARMS} \; d(snr)$

where snr is the signal-to-noise ratio and the limits of integration are over the expected operating range, probably 5 to ∞ dB. This equation implies that missed detections (Type II errors) and false alarms (Type I errors) will vary with snr, which is expected. The constant C implies that missed detections and false alarms might be weighted differently when measuring effectiveness. We have assumed that missed detections are considerably more serious than false alarms.

The task then, in summary, is to develop an alogrithm for signal/gap detection that minimizes $\emptyset$, which meeting the constraints of running time and implementing hardware.

THE APPROACH

It was noticed upon examination of typical speech and fsk data that the variance of the logrithms of the FFT magnitudes increases when signals are present in a channel. A detector designed to exploit this finding would have several desirable characteristics. First, variance is easy to compute, requiring only the summing then squaring and squaring then summing of values (about 3n operations). Second, the variance seems to be a fairly sensitive indicator of activity, much more sensitive than an energy level (mean value) detector. Third, use of the logrithm of the magnitude of the FFT makes the algorithm independent of time-invariant system gain levels. Note that:

$$\log(ax) = \log a + \log x = \text{constant} + \log x$$

But when considering the variance,

$$\text{variance}\,[\text{constant} + f(x)] = \text{variance}\,[f(x)]$$

This implies that

$$2) \quad \text{variance}\,[\log(ax)] = \text{variance}\,[\text{constant} + \log x] = \text{variance}\,[\log x]$$

Thus demonstrating the algorithm's independence of any constant gain a.

The use of logs complicates the algorithm, increasing its running time and preventing the calculation of the measure variance [log||FFT(signal + noise)||] as a function of the signal and noise distribution parameters.

THE ALGORITHM

The variance-based detection algorithm can be given as:

If variance [log ||FFT(waveform)||] > some threshold

declare activity.

Otherwise, declare inactivity.

The problem becomes establishing an appropriate threshold. Clearly, a low threshold will lead to a high false alarm rate, while a high threshold will cause missed detections. The threshold must be related to the variance of the noise alone as follows:

$$\text{threshold} > \text{variance}\,[\log ||\text{FFT}(\text{noise})||]$$

Hereafter, the right hand side above will be referred to as var(noise) for simplicity.

The communication channels under consideration are 4 kHz wide, but only the 300 - 3700 Hz band contains activity. This means that the first and last 300 Hz bands in the channel should contain only noise. So with a 64 point FFT, the middle 56 bins will contain the activity, if present and the first and last 4 bins will contain the FFT of the noise only. In practice, we have

noticed that the two bins immediately adjacent to the activity band on each side are contaminated due to filter and windowing imprecision. This leaves a total of 4 bins, 2 above and 2 below the activity channel, as measures of the noise only.

Four samples, however, are too few to get an adequate measure of the noise variance. If we can assume the noise to be ergodic, we can use the variance over time as a measure of the variance over frequency. That is, if the noise is ergodic, the statistical measures at any time over all frequencies will be the same as the measures at any frequency over all time. Because we have 4 noise bins, we will have 4 independent measures of noise variance over time. Given the assumption of ergodicity, these will be close to the variance of the noise over frequency and can be used in setting the threshold.

We experimented with different ways of using these 4 measures to arrive at a single estimate of var(noise). One method used, but discarded, is to take the maximum of the 4 values at any given time. The problem with this method is the extreme variability between successive threshold values. A second method, found to be more stable, is to take the average of these 4 variances as an estimate of the true variance. To establish a threshold, we increase this value by multiplying it by a "kluge" factor (K) to be determined empirically and depending upon the distribution of the noise over time and the trade-off between Type I and Type II errors. Thus,

3) threshold = K * average over 4 bins [var(noise) over time] > var(noise)

We can summarize the detection algorithm as:

If variance across inner 56 activity bins at any time [log ||FFT(waveform)||]

is greater than

K * average across the 4 outer bins {variance over time of each of the 4 bins [log ||FFT(waveform)||] }

declare activity,

Note again that the variance across the inner activity bins is at a single time window, while the variance of the noise is measured across all time windows.

In application, we are using a 64 point FFT requiring 128 data points. In general, activity will not truely begin or end at the start of one of these FFT cycles. Therefore, we expect a one cycle uncertainty regarding the true start and end of activity. A data sampling rate of 8000 for the 4000 Hz transmission channel requires 62.5 FFT cycles per second of 128 data points for real time processing. Thus, one cycle would translate to a 16 millisecond uncertainty regarding the true start and end of activity. Speech activity often begins and ends at low energy and, presumably, low variance levels, so the true uncertainty in practice would be even higher.

ALLOWING FOR TIME-VARYING NOISE DISTRIBUTIONS

We would like to make as few assumptions regarding the noise distribution as possible. We have thus far assumed only ergodicity. It may be possible to

create an algorithm which is robust against violations of this assumption by allowing for a slowly (with respect to the time of one FFT) time-varying noise distribution. This could be accomplished by using a moving, weighted measure of the variance across time of each of the 4 noise bins.

We will start by finding the moving, weighted mean of the magnitude of the FFT for each of the 4 noise bins.

$$4) \qquad X_n = \frac{a^0 x_n + a^1 x_{n-1} + a^2 x_{n-2} + \cdots\cdots\cdots a^n x_0}{a^0 + a^1 + a^2 \cdots\cdots\cdots a^n}$$

where x is the magnitude of the FFT of the i th bin at each time window and $o < a < 1$ is the attenuation constant.

For the purposes of calculation, this can be written recursively as

$$5) \qquad \tilde{X}_{n+1} = \frac{a * \tilde{X}_n + x_n}{d_{n+1}}$$

$$d_{n+1} = a * d_n + 1$$

Now we calculate the moving, weighted variance based on this weighted mean.

$$6) \quad VAR = \frac{a^0 (x_n - \tilde{X}_n)^2 + a^1 (x_{n-1} - \tilde{X}_{n-1})^2 + \cdots\cdots\cdots a^n (x_0 - \tilde{X}_0)^2}{a^0 + a^1 + a^2 + \cdots\cdots\cdots a^n}$$

This can be written recursively as

$$7) \quad U_{n+1} = a * U_n + (x_{n+1} - \tilde{X}_{n+1})^2$$

$$d_{n+1} = a * d_n + 1$$

$$VAR_{n+1} = \frac{U_{n+1}}{(n+1) * d_{n+1}}$$

The use of a moving, weighted variance, however, has effects not anticipated.

Basing the variance on a moving mean biases the measure, causing it to be an underestimate. There is a mechanical analogy illustrating this effect. If a pendulum is allowed to swing on a moving pivot, the pendulum will swing faster. The pivot moves in the direction of the center of gravity (moving mean). The faster swing indicates that the momemt of angular momentum (second moment or variance) has been lowered.

Even if the noise distribution were completely specified, finding the degree to which the variance is underestimated by this method would still be a formidable task. Recall from equation 3), however, that the average of the 4 bin variances will be multiplied by the kluge factor K. If the noise distribution doesn't vary too wildly, the kluge factor can be turned empirically to compensate for the underestimate of the variance.

It will be noted that the denominator in both the moving mean and variance calculations is a geometric series with a constant $0 < a < 1$. This series converges to $1/(1-a)$. For the purposes of implementing this algorithm into hardware, this limiting value is used through all calculations. This, of course, means that for the early cycles the mean will be grossly underestimated and the variance will be grossly overestimated.

GRACEFUL WARMUP

It was mentioned above that approximating the denominator of the moving statistics by the limit of the geometric series causes early measures to be very poor. There is a second reason to expect problems in the early cycles.

Note that in equation 5) the last term on the right hand side $(x_0 - \tilde{X}_0)$ is always zero because

$$X_0 = \frac{x_0}{1}$$

This means that VAR = 0 and, by equation 3), the threshold = 0. For small sample sizes (shortly after the algorithm is turned on) the variance, and hence the threshold, will be grossly underestimated. Consequently, early waveforms will invariably declared activity, whether or not activity is truely present.

This is compensated for in two ways. First, the algorithm can be given an adequate chance to warm up. In the computer program to be presented in following sections, the warm up period is interactively specified by the user. Second, if an upper bound on var(noise) is available a priori, it can be used to replace $(x - X)$ in equation 6). The following program also incorporates this method.

NOISE TRANSIENT AND ACTIVITY DROP-OUT FILTERING

We noticed in application that in the midst of a string of non-activity declarations, a spurious declaration of activity would often take place. In actuality, speech activity of single cycle duration would be most unusual. Similarly, single cycle drop-outs would often occur in the midst of a string of activity. To eliminate these problems, we adopted a "2 out of 3" filtering criterion. The filter is activated at the discretion of the program user by

interactively responding to query regarding filtering. The filter simply allows activity to be declared only if 2 of the last 3 cycles indicated activity. Similarly, non-activity is declared only if 2 of the last 3 cycles indicated non-activity.

One effect of the filter is to cause the first cycle of a string of activity to be invariably declared as non-activity. If real-time delay and storage requirements are not a problem, all data may be saved through the following cycle. If a cycle is the first in a string, the data may be retrieved after the next cycle and then written to the output files.

ESTABLISHING OPTIMAL PARAMETER VALUES

Parameters K, a, and the initial value of the var(noise) can be optimally set by exhaustion using equation 1). Due to the magnitude of this project, it has not yet been done. Rather, preliminary values of these parameters have been found which yield satisfactory results for the waveforms thus far tested.

Under the previous assumptions, the value of K should be greater than 1. Experimental evidence, however, has lead us to believe that for signal and noise environments thus far tested, K should be less than 1 for optimal performance. There are two reasons for this surprising result. The first is the judgment that Type II errors (missed detections) are more serious than Type I errors (false alarms). Recall that lowering the threshold decreases missed detections at the cost of false alarms. The second reason, however, is fundamentally related to the characteristics of the noise. We noticed that when activity is present, the magnitude of FFT's of the 4 outer bins drops about 40% for the waveforms tested. This would be true if an automatic gain control were present somewhere in the system. A time-varying gain causes the noise to be non-ergodic. The variances over time of the outer bin FFT's increase because the magnitudes change (decrease) whenever activity is present. All this serves to overestimate the noise variance. Thus, it appears that an optimal value of K will be somewhat below 1. In the following program, the user specifies K interactively.

The attenuation constant must also be specified, by nature, between 0 and 1. A low value allows greater movement of the weighted mean, thus lowering the measure of the variance and causing a high false alarm rate, which can be compensated for by raising the value of K. A high value causes the mean and variance to become more insensitive to changes in the noise parameters. This may be desirable as it causes the threshold to remain stable through brief periods of noise instability.

The parameters were chosen on the basis of tests on a 102400 value data set. This set yielded 800 FFT cycles. Approximately 258 of these cycles were identified ahead of time as containing activity. The remaining 552 were felt to be non-active. We found that for the single data set tested, with no added noise, the values K = 0.8, a = 0.997 and U = 57.0 worked well, yielding roughly 41 missed detections and 26 false alarms. For the purposes of speech compression, this result is entirely satisfactory.

THE DETECTION PROGRAM

A flow chart of the program is given as Figure 1. The program itself, written in the C programming language, is included in the appendix along with a variable

list. The main program requires four external functions (subroutines). These are: iread and iwrite, both written by Steven D. Beck to read into the main program the digitized signal values and to write to an output file the gap surpressed digitized signal; fft2, provided by Steven Beck, to perform the Fast Fourier Transform; and actgraf2, written by Walter Underwood to create a file for graphing on the line printer the indication of gap or activity.

NOISE SIMULATION

The program has the capability of adding noise to the signal prior to the variance calculations so that program performance may be tested at low signal-to-noise ratios. The program assumes that the warmup period consists of non-activity. The complex FFT data for each warmup cycle is stored. We exploit the linearity of Fourier transforms to add the warmup FFTs multiplied by an interactively set gain to the FFTs of subsequent cycles to get the equivalent FFTs of noisy data. The warmup FFTs are added to the subsequent FFTs cyclically.

The advantage of this method is that no assumptions regarding noise, except stationarity, are made. We are assuming, however, that the warmup period contains noise only. Consequently, the noise simulation works only with data files that begin with non-activity.

ALGORITHM PERFORMANCE

As previously mentioned, parameter values were found which yielded approximately a 16% missed detection rate and a 10% false alarm rate, basing both percentages on the number of cycles actually felt to contain activity. These values were found with data exhibiting approximately a 20 dB signal-to-noise ratio for single activity cycles. Added noise caused graceful performance degradation, primarily increasing the false alarm rate.

An obvious weakness of the algorithm, however, was its inability to detect the spoken word "six" as activity. The "s" and "x" sibilate sounds are broadband. The effect of these sounds is identical to adding noise to the activity band. The variance of the FFT of this band is not increased by the presence of these sounds, hence their undetectability. This appears to be a fundamental weakness of the algorithm and not the result of poorly chosen detection parameters.

HARDWARE IMPLEMENTATION

A flow chart indicating possible hardware implementation is attached.

SUMMARY

A detection algorithm has been written that gives good results for speech activity and also is simple enough for hardware implementation. Performance with fsk and psk signals has not yet been tested, but we expect satisfactory performance with these signal types as well. No assumptions regarding the distribution of the underlying noise are made, although alogrithm parameters are tuned on the basis of empirical knowledge of the noise. The system is robust against changing noise parameters and degrades gracefully with decreasing signal-to-noise ratio. The algorithm has proven unable to detect sibilates, but otherwise gives excellent performance.

APPENDIX

The following appendix contains the flow chart, the variable list and a listing of the program "vardet.c".

VARIABLE LIST

ang	added noise gain, used when simulating added noise
aten	atenuation constant for moving weighted mean and variance. 0 < aten < 1
avesnr	average signal-to-noise ratio over 6 declarations of activity
avevar	average of var(noise) across the s4 outer bins
counter	index used when windowing digitized waveform prior to FFT
count0 count1 count2	flags used for 2 out of 3 filtering. 1=activity, 0=inactivity. Count0 is the flag for the most recent FFT, count1 for the previous, count2 for second previous.
denom	denominator in weighted, moving mean and variance. Geometric series approximated by limit value 1/(1-a).
df	degrees of freedom of inner bin variance = # of bins - 1
end	flag transferred from iread(). 1= end of data
fin	general purpose character used for file names
fp fph fpw	file pointers
h[128]	coefficients for windowing waveform prior to FFT. read from file "FFTWIN".
i	general counter
idf	1/df
im	temporary variable name of imaginary part of waveform FFT plus imaginary part of added noise FFT
inoise[j][64]	imaginary part of the FFT of the noise. j th noise sample, i th frequency bin. IT IS ASSUMED THAT THE WARMUP PERIOD CONSISTS ENTIRELY OF NOISE.
irec	counter used exclusively by iwrite ()
isactive	indicator transferred to actgraf2() for graphing. 1=activity, 0=inactivity
k	counter for number of FFTs performed

klug	klug value used to multiply variance to establish threshold
mag[64]	magnitude of the FFT of the waveform plus added noise
n	number of data points in the FFT (128)
nar	interactively set flag to indicate whether continuous display of program variables k, varib, avevar, thresh and snr is desired. 1=desired.
nisact	counter for actual number of declarations of activity
nn	number of cycles indicating inactivity
ns	number of cycles indicating activity
r[64]	real part of the waveform FFT transferred from fft2()
re	temporary variable for real part of waveform FFT plus the real part of the added noise FFT
rnoise[j][64]	real part of the FFT of the noise. j th noise sample, i th frequency bin. IT IS ASSUMED THAT THE WARMUP PERIOD CONSISTS ENTIRELY OF NOISE.
snr	signal-to-noise ratio, calculated for each FFT
ss	sum of the squares of the logs of the magnitudes of the inner 56 activity bins, used to calculate the variance
ssn	energy in the outer 4 bins, used for calculating signal-to-noise ratio
sss	energy in the inner 56 activity bins, used for calculating signal-to-noise ratio
sum	sum of the logs of the magnitudes of the inner 56 activity bins, used to calculate the variance
suma[4]	the numerator in equation 5), used to calculate the moving, weighted mean
temp	log base 10 of the magnitude of the FFT bin currently under consideration.
thresh	threshold value for declaration of activity
two	interactively set flag to indicate whether 2 out of 3 criterion for activity declaration is desired. 1=desired.
Unext	U given in equation 7) used for calculating the $_{n+1}$ moving variance.

Uold	U_n given in equation 7) used for calculating the moving variance.
varib	variance of the inner 56 frequency bins
warmup	interactively set number of warmup FFTs
wtave	weighted, moving mean used to calculate moving variance
x[128]	digitized waveform data, read from input file
y[64]	imaginary part of the waveform FFT transferred from fft2()

```
/************************  PROGRAM: vardet.c ************************
 *
 * Written July, 1984 by Dr. J.L Wayman to do activity/gap detection.
 *
 * Program is written in the C language.
 *
 * Subroutines Required : iread(), fft2(), actgraf2(),iwrite()
 *
 */

#include <stdio.h>
#include <math.h>

#define PMODE 0644

short int fpr,fpw;

main()

{

float           r[256],
                y[256],
                h[128];

double          rnoise[16][64],inoise[16][64],
                mag[64],suma[4],
                Uold[4],Unext[4],
                sum,re,im,avevar,
                ss,avesnr,
                sss,ssn,snr,
                idf,df,ang,aten,
                denom,wtave,klug,
                varib,
                temp,thresh;

long int        two,nisact,
                nn,ns;

short int       x[128],
                n,
                counter,
                end,warmup,
                i,j,k,
                count0,count1,
                count2,
                nar,
                irec;

int             isactive;

char            fin[20];

FILE *fopen(), *fph;
FILE *fopen(), *fp;
```

```
/************************* OPEN I/O FILES ******************************/

if((fp = fopen("GRAF","w"))==NULL)
    {
     printf("Error cannot open GRAF\n");
     exit();
    }

if((fph = fopen("FFTWIN","r"))==NULL)
    {
    printf("\nERROR: cannot open %s for reading\n","FFTWIN");
    exit();
    }                                                    /* Read coefficients for
for (counter=0; counter<128; counter++)                  /* FFT windowing
    {
    fscanf(fph,"%f",&h[counter]);
    }

printf("\nEnter output file name:");
scanf("%s",fin);

if((fpw = creat(fin,PMODE))==-1)
    {
     printf("ERROR, cannot open output file for writing\n") ;
     exit();
    }

printf("\nEnter input filename : ");
scanf("%s",fin);

if((fpr=open(fin,0)) == -1)
   {
   printf("\nERROR: unable to open %s for reading\n",fin);
   exit();
   }

/********************  INTERACTIVELY SET PARAMETERS ********************/

printf("\nEnter added noise gain:");
scanf("%f",&ang);

printf("\nEnter attenuation factor for moving averages:");
scanf("%f",&aten);

printf("\nEnter kluge factor for threshold:");
scanf("%f",&klug);

printf("\nEnter warmup period (<=16):");
scanf("%d",&warmup);

printf("\nNarrative mode (1=yes , 0=no) ?");
scanf("%d",&nar);

printf("\nTwo out of three filtering (1=yes, 0=no) ?");
scanf("%d",&two);
```

```
/************************  INITIALIZE VARIABLES *********************/

n = 128;
irec=0;
df = 55.0;
end = 0;
idf = 1.0/df;
k = 0;
ns=0;
nn=0;
nisact=0;
denom =1/(1-aten);
isactive=0;
count0=0;
count1=0;
count2=0;

for(i=0;i<4;i++)
    {
    suma[i]=0;
    Uold[i]=57.0;
    }

/********************* READ AND PROCESS WAVEFORM DATA *****************/

while (1)                                   /* While there is still raw */
    {                                       /* waveform data to be read */
    iread(x,n,&end);
    if (end == 1) break;                    /* read and window data     */
    for (counter=0;counter<n;counter++)
        {
            r[counter] = x[counter]*h[counter];
            y[counter] = 0.0;
         }

    fft2(r,y,128,7,1);                      /* Go to subroutine for FFT */

    if(k<warmup)
       {
        for(i=0;i<64;i++)
          {                                 /*  During warmup period       */
           rnoise[k][i] = r[i];             /*  store away noise for later */
           inoise[k][i] = y[i];             /*  use in simulation          */
          }
       }

    for(i=0;i<64;i++)
       {
        j=k%warmup;
        re = r[i] + ang*rnoise[j][i];           /* Add noise to signal and */
        im = y[i] + ang*inoise[j][i];           /* take magnitude          */
        mag[i] = sqrt((double)(re*re +im*im));
       }

    sss = 0;                                /* These variables will be used   */
    ssn = 0;                                /* to establish snr for this cycle*/
```

```
/************************ COMPUTE ENSEMBLE NOISE VARIANCE *****************/

    for(i=0;i<2;i++)
        {
         temp=log10(mag[i]);                          /* First two FFT bins */
         suma[i] = aten*suma[i] + temp;
         wtave= suma[i]/denom;
         Unext[i] = Uold[i] + (temp*temp -2.0*temp*wtave + wtave*wtave);
         ssn = ssn + mag[i]*mag[i];

         temp = log10(mag[i+62]);                     /* Last two FFT bins */
         suma[i+2] = aten*suma[i+2] + temp;
         wtave = suma[i+2]/denom;
         Unext[i+2] = Uold[i+2] + (temp*temp -2.0*temp*wtave +wtave*wtave);
         ssn = ssn + mag[i+62]*mag[i+62];
        }

  /************ IF PAST THE WARMUP PERIOD, COMPUTE THE AVERAGE NOISE  ********
  ************ VARIANCE TO THIS TIME AND COMPUTE THE ACTIVITY BAND   ********
  ************************* VARIANCE FOR THIS CYCLE  ************************

  avevar =0;

  if(k>=warmup)
     {
       for(i=0;i<4;i++)                  /* Calculate average outer bin variance *
           avevar = avevar + 0.25*(k+1)* Unext[i]/(denom*k);

      thresh =(klug * avevar);                        /* Set the threshold */

      isactive=0;
      ss = 0;
      sum = 0;

      for (i=4;i<60;i++)
        {
        sss= sss + mag[i]*mag[i];              /* Calculate activity bin energy */
        temp = log10(mag[i]);                  /*for snr and find var of log mag*/
        ss = ss + temp*temp;
        sum = sum + temp;
        }

      varib = idf*(ss-sum*sum/(df+1.0));

      if(nar == 1)
        printf("\ncycle%d varib=%f avevar=%f thresh=%f",k,varib,avevar,thresh

/*************** COMPARE ACTIVITY BAND VARIANCE TO THE THRESHOLD ************

      if(varib > thresh)
        {
        count0=1;
        ns = ns + 1;
        if(two==1)
          {
```

```
          if(count0+count1+count2>=2)      /* If threshold is exceeded and   */
           {                               /* 2 of last 3 cycles also did so */
            isactive=1;                    /* declare activity and write raw */
            iwrite(x,128,&irec);           /* waveform data to output file   */
           }
         }

       else
         {
          isactive=1;
          printf("%d active\n",k);
          iwrite(x,128,&irec);
         }

       }
     else                                  /* If threshold is not exceeded */
       {                                   /* declare activity only in the */
        count0=0;                          /* event that 2 of the last 3 did*/
        nn = nn + 1;
        if(count1 + count2 == 2)
          {
          isactive=1.0;
          iwrite(x,128,&irec);
          }
       }
   }
                                          /* Dump activity status to file */
actgraf2(fp,isactive);                    /* for printing later           */

count2=count1;                            /* Update counters for 2 out of */
count1=count0;                            /* 3 criterion test             */

for(i=0;i<4;i++)                          /* Attenuate the numerator of   */
   Uold[i] = aten*Unext[i];               /* the moving, weighted variance*/

k++;                                      /* Count the number of cycles   */

/******************** CALCULATE SNR FOR THIS CYCLE **********************
******** SNR IS THE AVERAGE OVER EACH SIX DECLARATIONS OF ACTIVITY ******/

ssn = ssn/4;
sss = sss/56-ssn;

if(isactive==1)
  {
   if(nar==1)
      printf("\n**** ACTIVITY ****");
   if(nisact%6==0)
      avesnr = 0;
   if(sss>0)
     {
      snr = 10.0*log10(sss/ssn);          /* Calculate the average snr */
      avesnr = avesnr + snr/6;            /* over 6 declarations of    */
     }                                    /* activity and print out    */
   if(nisact%6==5 && nar==1)
      printf("ave snr= %f",avesnr);
```

```
   nisact++;
   }
                                        /* Go back to while statement,*/
}                                       /* go through another cycle   */
/****************** WHEN WE'RE ALL DONE, PRINT THE NUMBER  ****************
****************** OF CYCLES AND COMFIRM END INDICATOR    ****************
****************** FROM FILE READING SUBROUTINE           ***************/

printf("k=%d end=%d\n",k,end);

}
```

DISTRIBUTION LIST

DEFENSE TECHNICAL INFORMATION (2)
CENTER
CAMERON STATION
ALEXANDRIA, VIRGINIA 22214

LIBRARY, Code 0142 (2)
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

RESEARCH ADMINISTRATION (1)
Code 012
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

ADJUNCT PROFESSOR JAMES L. WAYMAN (15)
Code 53Ww
DEPARTMENT OF MATHEMATICS
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

DEPARTMENT OF MATHEMATICS (1)
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

PROFESSOR G. LATTA (1)
Code 53Lz
CHRMN, DEPARTMENT OF MATHEMATICS
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943